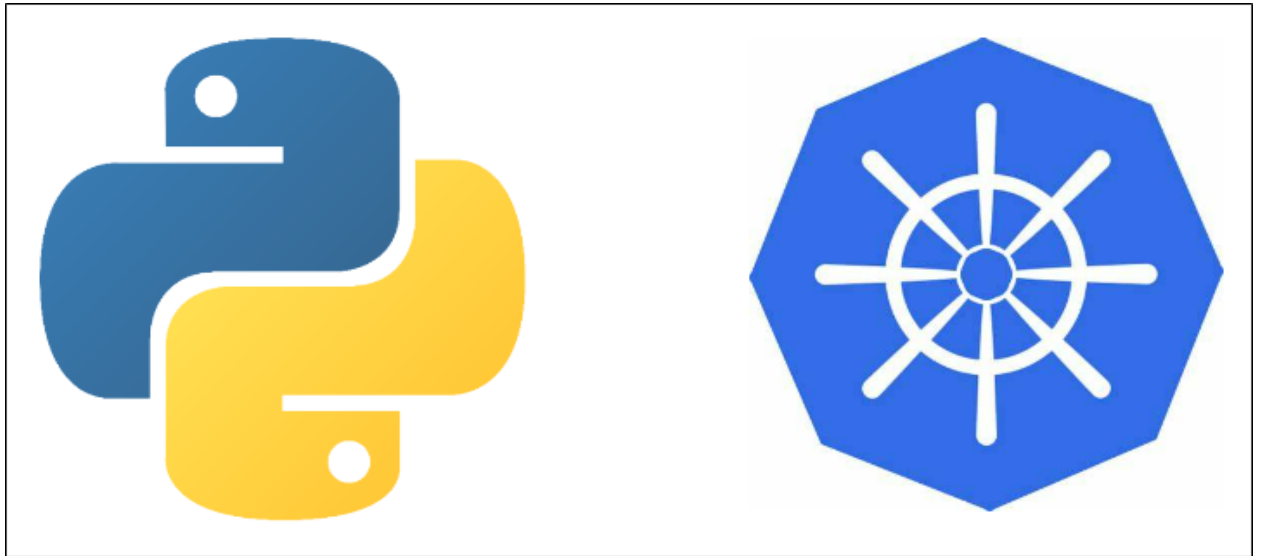


Python script for automating Kubernetes deployment



Share

In this blog, we will write a Python script to automate the deployment of Kubernetes resources using the [Kubernetes module](#). It is a Python client library. Using it we can interact with Kubernetes REST API to manage resources on our Kubernetes cluster.

To install the Kubernetes module use the following command.

```
pip3 install Kubernetes
```

Using the following python script we will create a Kubernetes object in a non-default namespace "dev". We have defined the Deployment object using a Python dictionary. You can specify deployments to make new ReplicaSets or to delete current deployments and replace them with new deployments that use all of their resources.

Our python script will take a unique ID as an argument so that we can create deployments with unique resource names.

```

try:
    ns = "dev"
    unique_id = sys.argv[1]
    deployment_name="dev-app-"+unique_id
    manifest = {
        "apiVersion": "apps/v1",
        "kind": "Deployment",
        "metadata": {"name": deployment_name},
        "spec": {"replicas": 1,
                  "selector": {
                      "matchLabels": {
                          "app": "dev"
                      }
                  },
                  "template": {"metadata": {"labels": {"app": "dev"}},
                                "spec": {"containers": [
                                    {"name": "webapp", "image": "nginx"}]
                                }
                  },
        }
    }

```

Next, we need to connect to Kubernetes API. We will use **config.load_kube_function()** to access the kubeconfig file. By default, it loads the kubeconfig file from \$HOME/.kube/config. We will manage the deployment object using **client.AppsV1Api** which allows us to work with all the resources that belong to **apiVersion: apps/v1**

```

config.load_kube_config()
v1 = client.AppsV1Api()

```

Now we will write a function to create a deployment object. It will also verify that the object is successfully deployed.

```
def create_deploy():
    deploy_names=[]
    list = v1.list_namespaced_deployment(namespace = ns)
    for i in list.items:
        deploy_names.append(i.metadata.name)

    if deployment_name in deploy_names:
        raise Exception("Invalid Argument: Resource already exists")
    else:
        deploy = v1.create_namespaced_deployment(body = manifest, namespace = ns)
        print("Waiting for Deployment to become ready...")
        time.sleep(90)
```



```
from kubernetes import client, config
import sys
import time
from logzero import logger
from chaoslib.exceptions import ActivityFailed
try:
    ns = "dev"
    unique_id = sys.argv[1]
    deployment_name="dev-app-"+unique_id
    manifest = {
        "apiVersion": "apps/v1",
        "kind": "Deployment",
        "metadata": {"name": deployment_name},
        "spec": {"replicas": 1,
                  "selector": {
                      "matchLabels": {
                          "app": "dev"
                      }
                  },
        "template": {"metadata": {"labels": {"app": "dev"}},
                      "spec": {"containers": [
```

```

        {"name": "webapp", "image": "nginx"}]
    }
    },
}
}
}

```

```

config.load_kube_config()
v1 = client.AppsV1Api()
field_selector =
"metadata.name={name}".format(name=deployment_name)
def create_deploy():
    deploy_names=[]
    list = v1.list_namespaced_deployment(namespace = ns)
    for i in list.items:
        deploy_names.append(i.metadata.name)

```

```

    if deployment_name in deploy_names:
        raise Exception("Invalid Argument: Resource already
exists")
    else:
        deploy = v1.create_namespaced_deployment(body = manifest,
namespace = ns)
        print("Waiting for Deployment to become ready...")
        time.sleep(90)

```

```

def deploy_status():
    ret = v1.list_namespaced_deployment(ns,
field_selector=field_selector)
    for d in ret.items:
        logger.debug("Deployment has '{s}' available
replicas".format(s=d.status.available_replicas))

```

```

        if d.status.available_replicas !=
d.spec.replicas:
            raise ActivityFailed( "Deployment '{name}'
failed".format(name=deployment_name))
        else:
            print("Deployment '{name}'
successful".format(name=deployment_name))

```

```

create_deploy()

```

```
    deploy_status()
except Exception as e:
    print(e)
```

Now we will execute our script to create this deployment object.

```
python deploy.py unique_id_001
```



```
(vagrant) (vagrant@test1 k8s)$ kubectl get all -n dev
NAME                                READY   STATUS    RESTARTS   AGE
pod/dev-app-unique-id-001-6d744c9ffb-w93bf   1/1     Running   0          2m45s

NAME                                READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/dev-app-unique-id-001   1/1     1            1          2m45s

NAME                                DESIRED   CURRENT   READY   AGE
replicaset.apps/dev-app-unique-id-001-6d744c9ffb   1         1         1       2m45s
```

[Please contact](#) our technical consultants if you have anything related to infrastructure to be discussed.